Artículo: Reinforcement Learning

GENERAL:

“Our aim is to evaluate the suitability of reinforcement learn-

ing [2], as a machine learning (ML) technique that enables

adaptive AI, for creating a learning agent in SC:BW. In order

to do this, we use two prominent temporal-difference (TD)

learning algorithms both in their simple one-step versions and

in their more complex versions that use eligibility traces. The

task chosen for this evaluation is managing a combat unit in a

small scale combat scenario in SC:BW. Eventually this agent

will be the part of a larger hybrid AI solution that addresses

the entire SC:BW gameplay problem, managing the different

layers of problems existing in a complex commercial RTS

game [3]. Our desire is for the technique to be easily scalable

for different types of units and even flexible enough to be

transferred to different problem domains such as other games

or even different sets of problems as described in [1].

”

“RL in games is a popular area of application for AI

research [4], with the technique being well suited to complex

game environments. Since Tesauro’s landmark use of TD

reinforcement learning [5], RL has been used in a variety of

different types of AI research both in conventional games and

in video games. RL has been used to create adaptive agents

in a fighting game [6], to control teams of players in the

commercial First-Person Shooter game Unreal Tournament

[7], to select city sites in the turn-based strategy game

Civilization IV [8] or, together with case-based reasoning

(CBR), to control sets of units in the RTS MadRTS in fighting

scenarios [9] to name but a few.”

NEURAL NETWORKS:

“use neural networks (NNs) to approach the problem of state

space complexity that is inherent in complex games such as

SC:BW. The NN receives the game state as input and then

approximates the state-action value function Q(st, at) to be

used for the Sarsa RL algorithm, thus creating neural-fitted

Sarsa (NFS). This technique is then shown with sufficient

training to significantly outperform the built-in game AI in

combat scenarios between a small number (3vs3) of similar

units and show promise in a larger scenario of 6vs6. However,

the authors also point out that the gains of NFS over normal

Sarsa were small. Scalability is also a problem as the learning

process was still very slow in the 6vs6 scenario, despite the

acceleration by transferring knowledge from the 3vs3 scenario.”

“Due to its enormous popularity and the well-documented

and well-maintained BWAPI interface, SC:BW has recently

seen a large increase in its use as a test bed for AI research.

Another reason for SC:BW’s popularity as a test bed for

research is the fact that it exhibits all the characteristics

identified by [3] as interesting AI problem areas in RTS

environments. These characteristics include adversarial

planning, incomplete information, spatial and temporal

reasoning, resource management, pathfinding and being a

multi-agent environment that generally allows for machine

learning and opponent modeling.

SC:BW has not only been used for research directly focusing

on the in-game AI but also for other topics such as to explore

its network traffic [12], for user identification through mouse

movement patterns [13] and procedural content generation to

automatically generate playable SC:BW maps [14].

Due to its large scope and complex problem domain,

StarCraft poses an ideal background for research in planning

algorithms.”

REINFORCEMENT LEARNING:

“Reinforcement learning is an unsupervised machine learning

technique which puts an agent into an unknown environment

giving it a set of possible actions and the aim of maximizing a

reward. As described in previous sections, RL has been applied

successfully to a wide range of problems from varying areas.”

“The agent in a reinforcement learning framework makes

its decisions based on the state s an environment is in at

any one time. If this state signal contains all the information

of present and past sensations it is said to have the Markov

property. If a RL task presents the Markov property it is said

to be a Markov decision process (MDP). A specific MDP is

defined by a quadruple (S, A,(Pss)^a, Rass). RL problems are

commonly modeled as MDPs, where S is the state space, A

is the action space, (Pss)^a are the transition probabilities and

Rass represents the expected reward, given a current state

s, an action a and the next state s0 . The MDP is used to

maximize a cumulative reward by deriving an optimal policy

π according to the given environment.”

EVALUATION AND RESULTS:

“For the evaluation of the performance of the selected RL algo-

rithms we designed a small scale combat scenario in SC:BW that

allows the RL agents to show their ability to learn in an unsupervised

environment. The scenario consists of one combat unit controlled by

the RL agent fighting a group of enemy units spread out around the

starting location of the RL agent as can be seen in figure 1. The RL

agent unit has the advantage of superior speed, superior range and - by

a small margin - superior firepower in comparison to a single enemy

unit. However, when fighting more than one enemy unit, the agent’s

single unit easily loses. Preliminary tests showed that using only the

built-in game AI, the single unit quickly loses in this scenario every

time. [...] An episode in our experiment concludes when either the

agent’s unit or all the enemy’s units have been eliminated.”

“One possible translation of the RL model into SC:BW

would have been to use a fixed number of SC:BW frames as one RL

time step. However, this proved problematic as the ’Fight’ action for

instance varies in duration between units and also depending on the

physical positions of units on the map. Therefore, we chose to define

one RL time step simply as the time it takes for that action to be

finished. For the ’Retreat’ action this meant that units would retreat

into the chosen direction at maximum speed for a fixed amount of

time.”

“We ran each of the RL algorithms for 1,000 episodes during which

the memory of the agent was not reset, i.e. after 1,000 games the

memory was wiped and the agent started learning from scratch. We

repeated these 1,000 episodes 100 times for each algorithm in order

to gain conclusive insights into their performance. For all algorithms

the agent followed an -greedy policy. However, the value of was

set to decline from its initial value of 0.9 to reach zero at the end

and thus make the action selection process more and more greedy

towards the end. [...]

After the 1,000-episode runs we repeated the process with shorter

500-episode runs, again with diminishing -greedy policy. This was

done in order to gain a better understanding of algorithm performance

in shorter terms and their speed of convergence towards an optimal

policy.”

Artículo: On Reinforcement Learning for Full-Lenght Game

GENERAL:

“StarCraft II poses a grand challenge for reinforcement learning.

The main difficulties include huge state space, varying

action space, long horizon, etc.”

“We investigate a hierarchical approach,

where the hierarchy involves two levels of abstraction. One is

the macro-actions extracted from expert’s demonstration tra-

jectories, which can reduce the action space in an order of

magnitude yet remain effective. The other is a two-layer hier-

archical architecture, which is modular and easy to scale. We

also investigate a curriculum transfer learning approach that

trains the agent from the simplest opponent to harder ones.”

“[...] reinforcement learning algorithms at present are still difficult

to be used in large-scale reinforcement learning problems.

Agents can not learn to solve problems as smartly and efficiently

as human. In order to improve the ability of reinforcement learning,

complex strategic games like StarCraft have become the perfect

simulation environments for many institutions such as DeepMind

(Vinyals et al. 2017), FAIR (Tian et al. 2017), and Alibaba (Peng et

al. 2017).”

“From the perspective of reinforcement learning, StarCraft

poses a grand challenge. Firstly, it is an imperfect informa-

tion game. Players can only see a small area of map through

a local camera and there is a fog of war in the game. Secondly,

the state space and action space of StarCraft are huge. [...] There

are hundreds of units and buildings, and each of them has

unique operations, making the action space extremely large.

Thirdly, a full-length game of StarCraft usually lasts from

30 minutes to more than an hour, and players need to make

thousands of decisions to win. Finally, StarCraft is a multi-

agent game. The combination of these issues makes Star-

Craft a great challenge for reinforcement learning.”

“Most previous agents in StarCraft are based on manual

rules and scripts. Some works related to reinforcement learn-

ing are usually about micromanagement (e.g. (Usunier et

al. 2016)) and macromanagement (e.g. (Justesen and Risi

2017)). These works solved some specific problems like lo-

cal combat in StarCraft. However, there are rare works about

the full-length games.”

REINFORCEMENT LEARNING:

*Leer estos párrafos para entender cómo funciona*

TRAINING ALGORITHM:

*Leer estos párrafos para entender cómo funciona*

REWARD DESIGN

“There are three types of rewards we explore in this paper.

Win/Loss reward is a ternary 1 (win) / 0 (tie) / -1 (loss) re-

ceived at the end of a game. Score reward is Blizzard score

get from the game engine. Mixture reward is our designed

reward function. [...] We have designed

reward functions for the sub-policies which combine dense

reward and sparse reward. These rewards seem to be really

effective for training, which will be shown in the experiment

section.”

EXPERIMENTS

“”